Containers overview

What are containers?

> A container is an isolated environment for your code. This means that a container has no knowledge of your operating system, or your files. It runs on the environment provided to you by Docker Desktop. Containers have everything that your code needs in order to run, down to a base operating system.

From Docker's website

Developers package their applications, frameworks, and libraries into a Docker container and ship those containers to be run outside their development environment.

Why use containers?

> Build, ship, and run anywhere.

Containers are self-contained and can run anywhere Docker can run. This means you can run a container on-premises or in the cloud, as well as in hybrid environments. Containers contain not only the application but also the dependencies; libraries and frameworks, configuration data, and certificates needed to run your application.

Here's a revised and more readable version of your text:

What are images

Docker images are immutable templates essential for creating containers. They ensure that applications operate consistently and reliably across different environments, which is vital for modern software development.

The creation of Docker images involves a process known as "Docker build". This proce
Dockerfile, a text document containing a sequence of commands. These commands a
essentially instructions guiding Docker on how to build the image.

Why use images?

Using Docker images is crucial in various stages of software development, including testing, development, and deployment. They play a key role in ensuring a seamless workflow across diverse computing environments.

What is Docker Hub?

After their creation, Docker images are stored in a registry, such as Docker Hub. From these registries, images can be easily downloaded and used to generate containers, facilitating widespread distribution and deployment of applications.

Now that you've got an understanding of Docker, containers, and images, let's move on to installing Docker.

Installing Docker

For this walkthrough, it is recommended to install Docker Desktop. Docker Desktop is bundled with a variety of tools including:

- Docker GUI
- Docker CLI
- Docker extensions
- Docker Compose

The majority of this walkthrough will be using the Docker CLI, but feel free to use the GUI if you prefer.

For the best installation experience, consult Docker's official documentation.

Running your first command

Now that Docker is installed, open a terminal window and run the following command:

```
docker version
```

You should see something similar to the following output.

```
docker version
Client: Docker Engine - Community
 Version:           24.0.7
 API version:       1.43
 Go version:        go1.21.3
 Git commit:        afdd53b4e3
 Built:             Thu Oct 26 07:06:42 2023
 OS/Arch:           darwin/arm64
 Context:           desktop-linux

Server: Docker Desktop 4.26.1 (131620)
 Engine:
  Version:          24.0.7
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.10
  Git commit:       311b9ff
  Built:            Thu Oct 26 09:08:15 2023
  OS/Arch:          linux/arm64
  Experimental:     false
 containerd:
  Version:          1.6.25
  GitCommit:       abcd
 runc:
  Version:          1.1.10
  GitCommit:        v1.1.10-0-g18a0cb0
 docker-init:
  Version:          0.19.0
```

If at any point you need help with a command, you can use the `--help` flag to provide documentation on the command you're running.

```
docker --help
```

Let's run `busybox` from the command line to print out today's date.

```
docker run busybox sh -c 'echo "The time is: $(date)"'
# The time is: Thu Jan 11 06:35:39 UTC 2024
```

- `busybox` is a lightweight Docker image with the bare minimum Linux utilities inst
  including `echo`
- The `echo` command prints the container's uptime.

Edit this page

Previous
« Containers

Next
Dockerfile »